Data-Driven Modelling and Simulation of Urban Transportation Systems Using CARMA

Data-driven modelling and simulation of urban transportation systems using CARMA

N. Zon[1] and S. Gilmore[1]

Laboratory for Foundations of Computer Science
School of Informatics, University of Edinburgh, Scotland
N.Zon@sms.ed.ac.uk, Stephen.Gilmore@ed.ac.uk

Abstract. Public transportation systems of different degrees and complexity are widely employed in cities around the world. Well-organised and efficient public transportation reduces traffic and the time spent commuting to work. In addition, more people choosing public transport rather than personal cars has a positive impact on reducing the number of vehicles on city roads: lessening their effect on climate change, improving air quality, and reducing noise pollution. Modelling and simulation of urban transportation systems is one way of analysing the influence that a variety of factors have on the overall functioning of the system. In this paper we present a Collective Adaptive Systems (CAS) model of an urban transportation system. We compare aspects of real data collected from a city bus system in the city of Edinburgh, UK, with the results of simulations of the CAS model constructed in the CARMA language. The simulations show results which are in good agreement with the real-world data, leading us to believe that the model could have useful predictive powers and thus provide an environment for experimentation with possible changes to the design of the system.

Keywords: Collective adaptive systems, urban transportation, stochastic modelling

1 Introduction

Modern urban transportation systems must be designed to have adaptive capabilities built-in because they need to respond to the unexpected events and circumstances which unfold as the delivery of the service progresses during the working day. This is particularly the case for bus services, where timetabled public transport must share the road network with private transport users who publish no timetable of their journeys and commuting plans and whose use of the road can depend on variables as diverse as the weather conditions, public holidays, and sporting events.

Set against this backdrop of hard-to-predict capacity availability of the underlying network, public transportation service providers must meet local or governmental requirements on quality-of-service as expressed through performance metrics such as percentiles of on-time departures or arrivals, excess waiting times, buses-per-hour requirements, and other measures [1]. In order to meet

quantitative targets such as these, public transport systems must have both *local* (point-of-view) and *global* (locus-of-control) adaptability, allowing system stakeholders to make both micro-scale service decisions (such as bus drivers speeding up, slowing down, or waiting at bus stops) and macro-scale decisions (such as shift operators re-routing buses, cancelling service instances, or deploying additional buses to cope with an unexpected surge in demand).

Human decision making is both in-the-loop within these systems, typically making locally autonomous micro-scale decisions, and outside-the-loop, typically making global macro-scale control decisions. Seen in this way, public transport systems can be viewed as *collective adaptive systems*, where (sometimes unexpected) behaviour emerges from the local interactions between actors in the system who are sharing resources when collaborating to meet common goals, even as they may be sometimes competing over resources in their efforts to satisfy individual priorities.

Modern smart transport systems are data-rich, making informed macro-scale decision making possible. Each vehicle in the fleet is equipped with GPS receivers and communications infrastructure to allow them to regularly report their location back to a vehicle tracking system. This *automatic vehicle location* (AVL) data provides anyone with access to the data with real-time oversight of the location of each vehicle in the fleet, making it possible to design applications which predict bus arrival times, and to compute metrics which provide statistical summaries of system performance in terms of key performance indicators which are of interest to system stakeholders.

Deeper insights into the causes of problems in service delivery can be obtained by combining data from several independent open data sources. This combination of data sets provides a different perspective on the use of the road network, allowing us to make a more detailed model which would not be possible if working from a single source of data. For the model in this paper, we have combined AVL data which was obtained from the Transport for Edinburgh company [10] with long-run average data on traffic intensity from the Tom Tom satellite navigation service [13]. These are two genuinely independent data sources, the real-time vehicle location system on the buses does not provide data to the Tom Tom network, which harvests data from their own propriety hardware installed in private vehicles.

In order to analyse a real-world example of a problem in this domain, we constructed a formal model of the system of interest in the modelling language CARMA (Collective Adaptive Resource-sharing Markovian Agents) [5,6] and studied it via simulation. In contrast to logic-based explicit state-space analysis approaches, such as probabilistic model-checking [2] with model-checkers such as Storm [3] and PRISM [4], simulation provides no absolute guarantees of correct behaviour but it scales to allow the construction of very detailed spatial models of systems such as the location-accurate bus route which we have modelled here. In contrast to blended approaches such as statistical model-checking, we have found using simulation directly with CARMA to be a more effective tool for communication of results to system stakeholders, largely because of the

absence of difficult-to-understand logical characterisations of system properties, and the avoidance of the attendant formal machinery of verification-based approaches.

2 The CARMA language

At its core, CARMA is a value-passing process calculus which supports both attribute-based broadcast communication and unicast communication between agents. Agents have defining *attributes* and a local store which records their current state. Agents can perform *activities* either individually, or through unicast cooperation with another agent, or through broadcast cooperation with a collective of agents which is formed dynamically at run-time via attribute-based selection. Broadcast activities are decorated with a star suffix (as in $move^*$) to distinguish them from unicast activities. Agent instances can be both created and destroyed as the simulation unfolds.

CARMA builds on a long tradition of work on stochastic process calculi and incorporates ideas from the ensemble-based modelling language SCEL [7], which in turn inspired the development of the core calculus AbC [8] which focuses on a minimal set of primitives to describe attribute-based communication.

All activities in CARMA have an associated *rate* which limits how frequently they can occur as time elapses during the simulation execution. Rates can be defined by arbitrary functions which allow us to model detailed distributions of events as determined from measurement data and other sources. Probabilistic execution of activities can also be specified.

CARMA models are executed via the CARMA *Eclipse Plug-in* [9], a fully-featured development environment for CARMA which provides project structure for CARMA projects, a syntax-aware editor which is fully integrated into the Eclipse platform, efficient simulators for the language, and a reporting framework to provide plotting capabilities for results within the modelling platform itself.

In addition to these analysis tools, the CARMA Eclipse Plug-in also provides a data modelling layer on top of the CARMA process calculus. This language, CaSL, (CARMA *Specification Language*), supplements the core process calculus with data types such as integer, real and boolean, and data structures such as arrays and records. The addition of this data modelling layer on top of the core process calculus facilitates type-checking and static analysis of models which leads to detection and correction of errors in models in a way which is not possible in untyped process calculi.

The modelling advantages of having this additional expressive power in the language do not necessarily become obvious when working with smaller example models, but its benefits become clear when working with larger and more complex models, especially those which need to be maintained and updated over time. We have found this facility to be invaluable for catching modelling errors on several occasions and we believe that the added clarity which the model has through the use of data types and data structures makes it easier to explain

to other stakeholders in the project, some of whom are not experts in process calculi and formal languages.

3 CARMA model

3.1 Locations

Coordinate system In our model, we use locations extracted from the real data provided to us by Transport for Edinburgh. The Transport for Edinburgh API represents a location by its geographical latitude and longitude. In order to reduce computation time when working with location datasets, we decided to translate these to the Universal Transverse Mercator (UTM) system. In this system a location is given by the tuple (easting, northing, UTM zone number, UTM zone letter). Easting and northing coordinates are equivalent to x and y coordinates on a plane, and the origins of the x and y axes depend on the UTM zone number. In our model we are considering only the city of Edinburgh. It lies in a single UTM zone (30U). This means that we do not need to take the zone number into account when performing computations. The easting and northing values give us enough information to process the data.

In the rest of this paper we refer to the easting and northing values as the x and y coordinates. (It should be noted that when applying this modelling technique to a different city one needs to take into account the possibility that the considered area lies in more than one UTM zone.)

The easting and northing coordinates are expressed in the units of metres, and so are all the distances in the presented paper, unless stated otherwise.

Data types Each location in the model is represented by the tuple `(id,x,y)`. The `id` of a location is a unique identifier of a given location.

In the data from the Transport for Edinburgh API, locations are represented using latitude and longitude values from a continuous domain, as the buses move continuously and a given bus can be found using GPS at any location at the time of sampling. In our model, we represent movement as sequences of steps between discrete locations. These consist of locations of bus stops as well as any number of points on the way between two adjacent stops.

Looking up data inside the model The location data within the model is usually requested in the same sequence as a bus would traverse these points on its journey. For example, if a given model contains only those locations that represent a stop, each Bus entity would attempt to look them up one by one in the order in which they appear on that particular service's timetable. For this reason we map the original bus stop unique identifiers from the API sourced data into a different set of unique identifiers associated with a particular CARMA model name-space domain. In this way, we can ensure that for each service, the identifiers of locations which the bus is due to traverse, are represented by incrementing integers.

This means that, conveniently, if a given bus is currently at the location with `id==i`, its next location has the identifier `id==i+1`.

Records in CARMA are indexed, and in our model the index represents the id of a given location. There are two records, x and y, for storing the x- and y-coordinate values of each location.

3.2 Departures

In our model, new Bus components need to be instantiated with a rate that reflects the departures of buses from the initial stop in the real data. A common pattern shared by the timetables shows buses departing less frequently in the morning and evening hours, and more frequently during the day, however the exact pattern of departures differs from service to service.

In the CARMA system, we model the departure timetables using a function which returns a transition rate for triggering an action that results in instantiating a new Bus component at a starting location. This rate should be dependent on the time of the day in a way that reflects the data in the timetables.

In order to obtain the rate of transition for the Bus component instantiation, we will use a function that returns the period of bus departure occurrences given a certain time of the day. This function is calculated from the list of departure times on a timetable (there is a timetable for every given day of the week, service and destination).

One way to calculate an approximate value of the period of bus departure events, is to look at the differences between consecutive bus departures in a certain range of time around the time for which the period is being calculated. For example, when calculating the frequency of bus departures at 12 am, we take a mean of the time differences between consecutive buses departing between 9 am and 3 pm.

Another approach is to calculate a weighted average of the differences between bus departure times, with the weight being proportional to the time difference between the considered departures and the time of sampling. For example, when calculating the frequency of bus departures at 12 am, we take into account all of the time differences between consecutive buses that can be calculated from the available data. However the closer their time is to 12 am, the higher the weight associated with their value when calculating the average. In other words, this formula is a weighted average of differences between consecutive bus departure events and the weight is the distance [in the unit of time] between the average value of the two departures whose difference we are calculating, and the point of sampling.

The comparison of the two methods described above is presented in Fig. 1. The graph is based on the timetables of two stops which are the starting locations of service number 5 to Hunter's Tryst. The early morning departures start at Brunstane, while the rest of departures start at The Jewel. The vertical lines represent actual departure times, as sampled from the timetables.

In the simulation results presented later in this paper, we used a model based on the period function generated using the weighted average method.

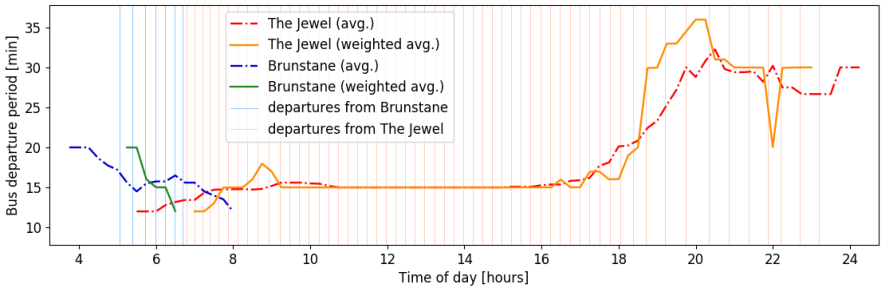

Fig. 1: The period of bus departures per time of day calculated using a simple average over 3 hours, and using the weighted average method.

3.3 Movement

Changing locations In the presented model, a Bus is a CARMA component having two states, "ON ROAD" and "AT STOP". The state graph representing its behaviour is shown in Fig. 2.

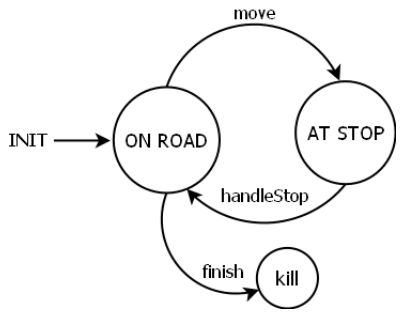

Fig. 2: The internal state diagram of the Bus component.

In the "ON ROAD" state, the Bus component can perform a transition which, depending on the value of the predicate expression, has two possible outcomes:

- the Bus component updates its store to reflect its new location; or
- the Bus component is removed from the system.

In the CARMA model this is represented by two actions, "move*" and "finish*". Performing either of them results in exiting the "ON ROAD" state. If the "move*" action is triggered, the bus enters the "AT STOP" state. In the other case, the bus has reached its destination, and the component is removed from the system.

In this manner, each of the Bus components traverses a number of locations on its route, until arriving at the destination. The continuous nature of movement of real-world vehicles is modelled using this step-based discrete approach.

When in the state AT STOP, the Bus component needs to perform the handleStop action in order to transition to the ON ROAD state. This action simulates the additional waiting time a bus takes when passengers are leaving and boarding. This transition's rate depends on the kind of location the bus is currently at. When in this state, the bus component can be in one of the two situations:

- the current location in the store is a stop
- the current location in the store is a point on the route but not a stop.

If the location is just a point on the route and not a bus stop, the action has a fast rate, which means its delay time can be ignored and we may assume that it is triggered instantaneously. In the other case, the rate has a constant value which reflects the waiting time.

Speed The average speed at which a bus is moving is modelled by adjusting the rates of the "move*" and "handleStop*" actions of the Bus component in the CARMA system.

The rate of the "move*" action depends on the distance between the current and next location as well as the current value of traffic. The "handleStop*" action has a constant rate, but it only results in delays in locations which are stops.

3.4 Limitations of the CARMA implementation

Because of the limitations that the syntax of CASL imposes on defining measures, the data obtained for the analysis couldn't be extracted directly from the CARMA simulation in the usual way. Each CARMA model for the purpose of simulation is translated to the Java programming language. This is done automatically, each time a CARMA file is saved in the Eclipse IDE.

One set of data that is not directly available through CASL, is the exact values of store variables of all components in the system, at any given time. In CASL, it is only possible to define measures that return the average, minimum or maximum from the set of sampled values. In order to obtain the complete data, sampled at each step of the simulation, the generated Java project was post-processed to save the following information, after each Bus component performs an action:

- bus identifier
- current time
- bus start time
- bus current location (x, y)

This post-processing was performed automatically using Python scripts (available for download at [12])

4 Application of the model

4.1 Real world data source

The Transport for Edinburgh company [10] has provided us with access to the REST API that can be used to obtain data gathered from their bus system. The data used for the purpose of this research is relevant to the routes, timetables and live vehicle locations of bus services provided by Lothian Buses [11], a bus operator within the Transport for Edinburgh company. The data consists of JSON-formatted text files which can be requested from the following endpoints:

- **stops** (the latest information on bus and trams stops served by Transport for Edinburgh [10], including fields such as name of the stop, available services, destinations, and its geographical location)
- **services** (the information on each available service's name, destination, routes, and stops)
- **timetables** (the full timetable information per stop, including the list of services with corresponding departure times)
- **journeys** (the list of expected stops and corresponding departure times for a given service leaving from a particular start stop at a particular time and arriving at a particular destination point)
- **stop-to-stop timetables** (timetables in the form of journeys between two stops which share a route, for a requested time)
- **service status** (up-to-the-minute information about disruptions affecting services in real time)
- **live vehicle locations** (real time information on the position of currently active vehicles)

The comprehensive documentation can be found at `https://tfe-opendata.readme.io/`.

Locations and points on routes The objects ascribed location values are either bus stops or points on the route. The points on the route are included to preserve the shape of the bus route, which, when reconstructed using only stop locations may be missing important information such as road turns. In this paper the term "points on routes" is used to collectively refer to bus stops as well as other points the route whose locations are included in the Transport for Edinburgh API data. In the API data, each location is represented by its geographical latitude and longitude.

Bus stops Apart from their locations, bus stops have been additionally given a unique identifier and a name, for example "Shandwick Place".

Services The `services` API endpoint provides information about all the existing services - including their names, destinations, and routes.

Live vehicle locations The live vehicle locations endpoint of the Transport for Edinburgh Open Data API provides snapshots of the bus system at the time of request. This information includes the geographical location of all the currently active buses, as well as the next stop on their journey.

4.2 Traffic

The Tom Tom Traffic Flow [13] service is a service provided by the Tom Tom company, dedicated to providing traffic information in a number of large cities around the world. The service offers live traffic data as well as historic traffic data gathered from Tom Tom GPS devices in vehicles on roads.

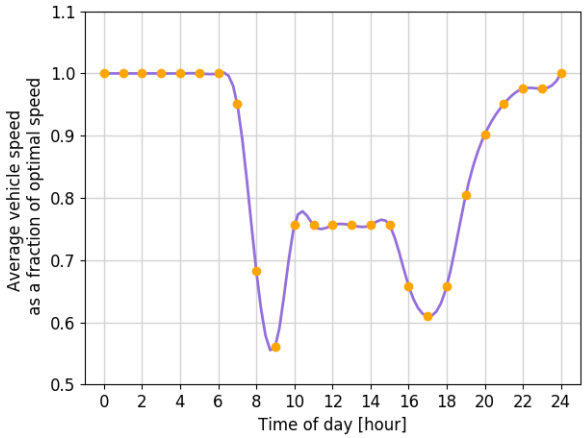

Fig. 3: The average speed of a vehicle in The City of Edinburgh, expressed as a fraction of the optimal speed. The data shows a congestion peak at 9 am and a smaller congestion peak at 5 pm. On that particular day, the optimal vehicle speed was equal to 25.5 mph (41 km/h), represented by 1.0 on the graph above.

The traffic data is available in the following three formats:

- Live Traffic Level: a measure of traffic level per time, expressed as a percentage, and described on the service website [13] [14] as "Indication of the current severity of traffic congestion on monitored roads in the city area compared to the normal expected congestion level. Includes highways, major roads and minor roads."
- Live Traffic Speed: a measure of the average vehicle speed in the unit of km/h, per time. The service also provides a value of "optimal speed", that is the average speed of vehicles when no traffic is present.

– Live Traffic Reports: provides information on the causes of increased congestion, classified into three categories: roadworks, jams and closures.

In our model, we used the data from the Live Traffic Speed service, as presented in Fig. 3. To lessen the probability that an incident (roadwork, closure) not included in the simulation would affect the results, we used data from a day in which all the causes of increased congestion were categorised as "jams" in the Live Traffic Reports service.

4.3 Instance of the model: Service number 5 to Hunter's Tryst

In order to create an instance of the model, we used data collected from buses on the route of Lothian Buses' service number 5 to Hunter's Tryst, using the weekday timetables (that is, excluding Saturdays, Sundays and Bank Holidays).

This route has 124 points on routes, 58 of which are stops.

We applied the same measures to the data obtained from the Lothian Buses API and from the simulation of our Carma model.

We compare the real-world data with the model using the number of active buses per time of day. This measure depends on two aspects of the system:

– the frequency at which new buses are introduced into the system
– the length of time a bus remains in the system.

In this case, both of these characteristics are time-dependent and at the same time independent of each other.

The results obtained from the simulation correspond well to the real-world data. The greatest disagreement can be observed in the initial stage of the system evolution (morning hours). This discrepancy may be explained by the artefacts of the real data: many vehicles that are to start their service only later in the day appear as active in the system as soon as the GPS mechanism is switched on. In the presented graphs we removed the buses that appear as active before the first departure time of the timetable, however further and more selective data cleaning needs to be performed to eliminate this inconsistency in the later morning hours.

An interesting observation can be made about the influence of the traffic on the number of active buses. In the simulation instance without traffic (i.e. the speed depends only on the distances), shown in dark blue in Fig. 4, the number of active buses is underestimated in the time ranges 9:00 am - 11:30 am, 4:00 pm - 9:00 pm and overestimated between 2:00 pm and 3:00 pm. In the simulation with traffic (shown in dark red), the trend seems to be reversed for the time ranges 9:00 am - 11:30 am and 2:00 pm - 3:00 pm. This means that if the influence of traffic was smaller by a particular amount, the simulation would fit the data with greater accuracy. The reason for the traffic to have a smaller influence on buses, than it has on other vehicles (those equipped with Tom Tom GPS devices, which are the source of the traffic information) is probably the fact that buses can travel along privileged bus lanes. If we assume that a majority of vehicles used by Tom Tom for data gathering do not travel in such lanes, this

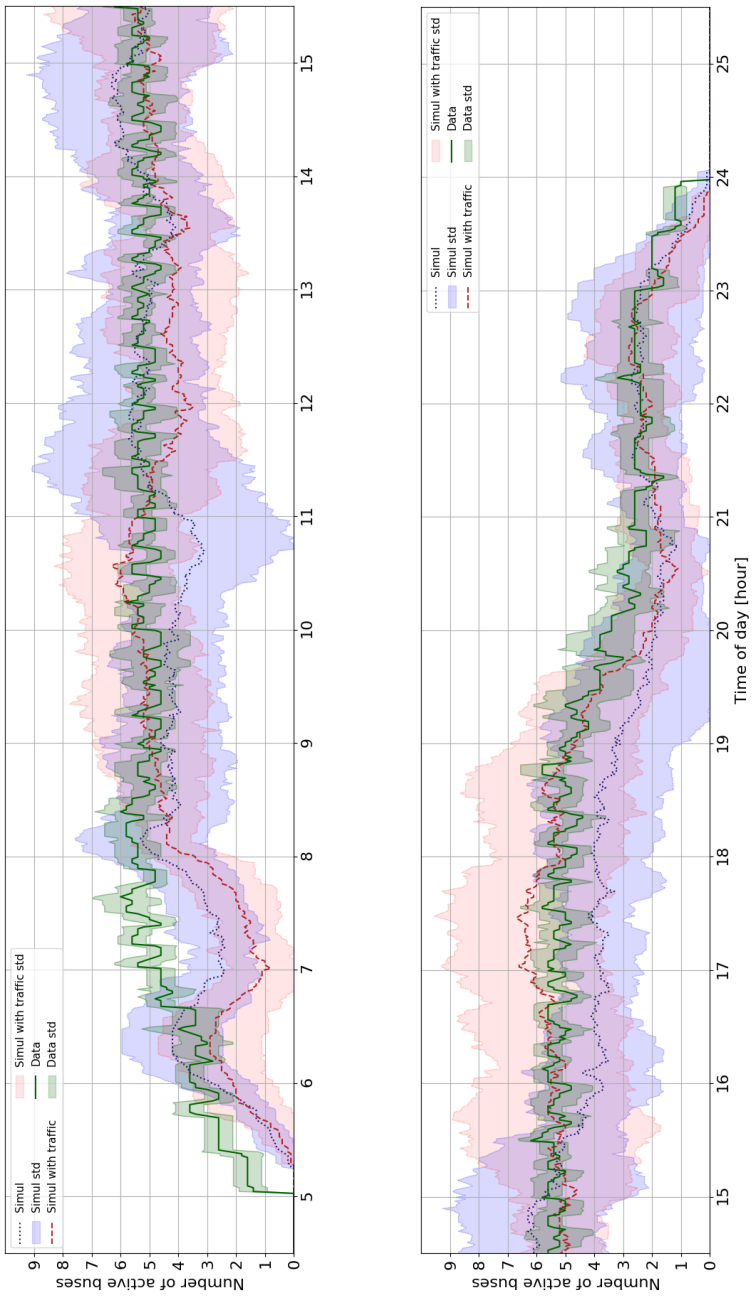

Fig. 4: The number of active buses versus the time of day, observed in the real data, simulation with traffic and simulation without traffic.

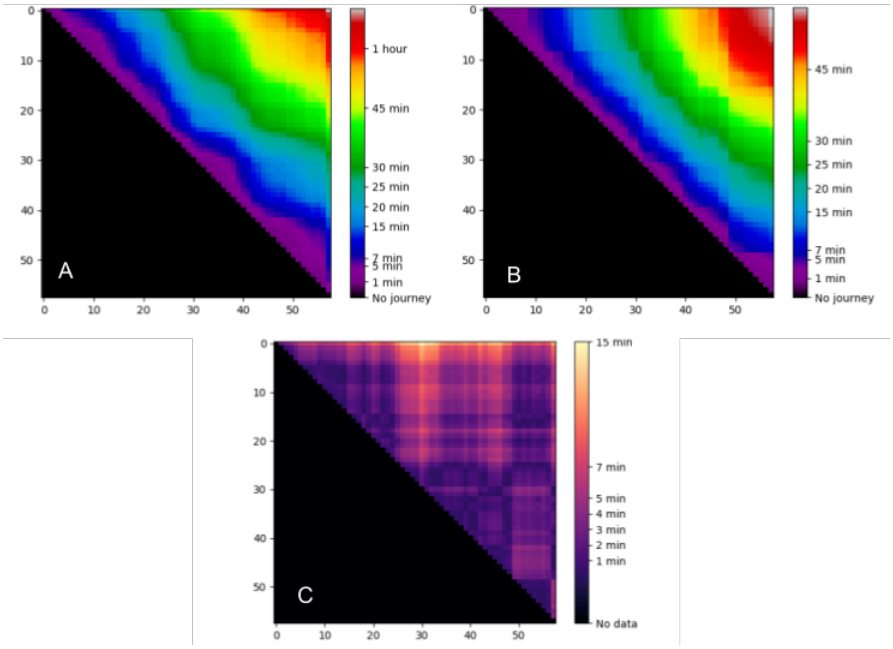

Fig. 5: The average journey length for each pair of bus stops on the route arriving at the destination stop between 5:30 pm and 6:30 pm. Panel (A): real data, panel (B): simulation, panel (C): absolute difference between data and simulation.

discrepancy can be explained by the lower average speed of those vehicles than that of buses.

Between 7:45 pm and 12:00 am the inclusion of traffic seems to have a negligible effect on the simulation, and the active buses count is underestimated by both simulation instances in the time range 7:30 pm - 9:30 pm.

In the last time segment, both simulation instances seem to align with the real-data results very well.

An hour long snapshot of the system taken between 5:30 pm and 6:30 pm can be seen in Fig. 5. The x and y axes represent bus stops, which are indexed with integers, and sorted by the order they appear along the route. The colour of each pixel represents the average journey time from the stop designated by the x-axis value to the stop designated by the y-axis value. The striped pattern of discontinuities, which can be observed on all plots, are more pronounced in the simulation results. This is because they are the results of accumulated delayed or early departures of a bus from consecutive bus stops on the journey. In real life, when a bus arrives at the stop too early, the driver waits until the timetable departure time before continuing, minimising the overall value of headway. In this simulated model that mechanism was not represented. For this reason, we can observe that on some journeys, the duration times have been shifted for all consecutive arrival stops.

5 Conclusions

In this paper we applied the recently-created Carma process calculus to model an urban transportation systems. The challenges of this technique originate from the continuous nature of the real-world data being represented in a discrete and stochastic modelling environment. The patterns one might expect to observe when viewing the system from a high-level and simplified perspective, are often distorted by numerous factors that influence the real-world data. For this reason, a stochastic model will always be an abstract interpretation of the considered system. The challenge therefore is to extract a generic paradigm that can describe a given instance with a satisfiable accuracy, while at the same time remaining applicable to a range of other instances.

The simulations based on the presented modelling approach result in patterns which are in good agreement with those observed in the real data. An interesting feature of the Transport for Edinburgh transportation system which emerged from comparing the data with simulation, is the fact that traffic influences buses to a lesser extent than other vehicles (presumably because of the existence of bus lanes).

This good agreement between simulation results produced by the model and real-world behaviour means that experiments with the model can be used to effectively evaluate potential modifications to the real-world system, or to check the accuracy of other descriptions, as in [15].

References

1. Daniël Reijsbergen and Stephen Gilmore (2014) *Formal Punctuality Analysis of Frequent Bus Services Using Headway Data.* In: Horváth A., Wolter K. (eds) Computer Performance Engineering (EPEW'14). Lecture Notes in Computer Science, vol 8721. Springer, Cham.
2. Joost-Pieter Katoen (2016) *The Probabilistic Model Checking Landscape.* In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'16). Pages 31–45.
3. Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen and Matthias Volk. (2017) *A Storm is Coming: A Modern Probabilistic Model Checker.* Proceedings of the 29th International Conference on Computer-Aided Verification (CAV'17), Volume 10427 of LNCS, pages 592–600, Springer.
4. Marta Kwiatkowska, Gethin Norman and David Parker. (2011) *PRISM 4.0: Verification of Probabilistic Real-time Systems.* In Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585–591, Springer.
5. Luca Bortolussi, Rocco De Nicola, Vashti Galpin, Stephen Gilmore, Jane Hillston, Diego Latella, Michele Loreti and Mieke Massink. (2015) *CARMA: Collective Adaptive Resource-sharing Markovian Agents.* Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems, (QAPL'15), EPTCS 194, pages 16–31.
6. Michele Loreti and Jane Hillston (2016). *Modelling and Analysis of Collective Adaptive Systems with CARMA and its Tools.* Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems – 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2016, Bertinoro, Italy, June 20-24, 2016, Advanced Lectures. Lecture Notes in Computer Science 9700, Springer. pages 83–119.
7. Rocco De Nicola, Michele Loreti, Rosario Pugliese and Francesco Tiezzi (2014). *A Formal Approach to Autonomic Systems Programming: The SCEL Language.* ACM Transactions on Autonomous and Adaptive Systems (TAAS). Volume 9 Issue 2, July 2014.
8. Yehia Abd Alrahman, Rocco De Nicola and Michele Loreti (2016). *On the Power of Attribute-Based Communication*, In: Albert E., Lanese I. (eds) Formal Techniques for Distributed Objects, Components, and Systems. FORTE 2016. Lecture Notes in Computer Science, vol 9688. Springer, Cham.
9. Jane Hillston and Michele Loreti (2016). *CARMA Eclipse Plug-in: A Tool Supporting Design and Analysis of Collective Adaptive Systems.* Proceedings of the 13th International Conference on Quantitative Evaluation of Systems (QEST'16), Lecture Notes in Computer Science 9826, Springer. pages 167–171.
10. Transport for Edinburgh Ltd., `http://transportforedinburgh.com/`
11. Lothian Buses, `https://lothianbuses.co.uk/`
12. GitHub repository of scripts used for data handling, `https://github.com/nataliazon/LothianBusesScripts/`
13. Tom Tom Traffic Flow for Edinburgh, `https://www.tomtom.com/en_gb/traffic-news/edinburgh-traffic/traffic-flow/`
14. Tom Tom Traffic News, `https://www.tomtom.com/en_gb/traffic-news/`
15. Ludovica Luisa Vissat, Allan Clark and Stephen Gilmore (2015). Finding Optimal Timetables for Edinburgh Bus Routes, *Electronic Notes in Theoretical Computer Science*, Volume 310, pp. 179-199.